International Journal of Engineering and Applied Computer Science (IJEACS)
Volume: 02, Issue: 09, ISBN: 978-1-78808-411-6, November 2017

DCHEFT Approach for Task Scheduling to Efficient Resource Allocation in Cloud Computing

Mahendra Bhatu Gawali
Department of IT, Thadomal Shahani Engineering College, Bandra(W), University of Mumbai, Mumbai, MS, India.

Subhash K. Shinde
CSE Dept, Lokmanya Tilak College of Engg. Koparkhairane, Navi Mumbai
University of Mumbai, India.

***Abstract*- Task scheduling is an important aspect to improve the utilization of resources in the Cloud Computing. This paper proposes a Divide and Conquer based approach for heterogeneous earliest finish time algorithm. The proposed system works in two phases. In the first phase it assigns the ranks to the incoming tasks with respect to size of it. In the second phase, we properly assign and manage the task to the virtual machine with the consideration of ideal time of respective virtual machine. This helps to get more effective resource utilization in Cloud Computing. The experimental results using Cybershake Scientific Workflow shows that the proposed Divide and Conquer HEFT performs better than HEFT in terms of task's finish time and response time. The result obtained by experimentally demonstrate that the proposed DCHEFT performance superiorly.**

Keywords- Cloud Computing; Task Scheduling; Resource Allocation; Divide and Conquer; HEFT.

I. INTRODUCTION

The Cloud Computing is an emergent technology now a days, which can support executing computationally heavy weight application to infrastructure capital issues. The Cloud Computing has overcome on the traditional way to deliver service offline. The Cloud can serve the customer by providing various types of services like SaaS, PaaS, IaaS [1]. The popularity of Cloud Computing has continuously increasing with respect to advanced technology. Both academia and industry are shifting their traditional infrastructure setup to Cloud and being started to provide the services in public, private and hybrid type of Cloud. Now days, there are number of service providing companies are available in the technology market which offers the Cloud Services [2,3,4]. To choose the service provider is completely depends upon the type of service user wants. As per the requirement the user takes a decision about certain services to be taken from certain service provider. More or less this condition may applicable to all the Cloud Service User and Cloud Service Provider. In the general different users may submit their respective different task with various demands of resources. To fulfil the demands of every user request with the available resources is the challenge for Cloud Computing. It may also effects on the performance of the Cloud Computing. The objective function of this problem is to schedule the tasks on resources and manage their executions so that execution time for the task will decrease and resources also utilized properly. Basically, task scheduling is quite complex and important aspect. In this work we extend the heterogeneous earliest finish time [5] methodology by introducing the divide and conquer approach in it. Google Compute Engine used the Cron tool for scheduling the tasks [6]. The major contribution of this paper summarized as follows.

1) We modify the ranking algorithm which will assign the ranks to the incoming Cloud requests.
2) Divide and Conquer methodology has been added into heterogeneous earliest finish time for scheduling.
3) The experimental performance of the proposed solution using Cloud Simulator.

The remainder of this paper is organized as follows. The section II will brief about the state-of-the- art in the Cloud Computing. The task-scheduling problem is described in section III. The section IV explains the divide and conquers approach for heterogeneous earliest finish time. The section V illustrates the experimental setup of the proposed DCHEFT in addition with this its gives performance evaluation with existing algorithms. Finally, the section VI concludes the paper.

II. LITERATURE SURVEY

This section will brief about the state-of-the-art in the various algorithms used to solve task scheduling issues in the Cloud Computing.

Liu et al. have been designed a model for a programming, which utilized the large scale data intensive batch applications [7]. It can specify the data partitioning and the computation task distribution, while the complexity of parallel programming is hidden. Fallenbeck et al. present a dynamic approach to create virtual clusters to deal with the conflict between parallel and serial jobs [8]. In this approach, the job load is adjusted automatically without running time prediction.

Wilde et al. proposed Swift, a scripting language for distributed computing [9]. Swift focuses on the concurrent execution, composition, and coordination of large scale independent computational tasks. A workload balancing mechanism with adaptive scheduling algorithms is implemented in Swift, based on the availability of resources. A dynamic scoring system is designed to provide an empirically measured estimate of a site's ability to bear load, which is similar to the feedback information mechanism proposed in our design. However, the score in the Swift is decreased only when the site fails to execute the job.

Junjie proposed a load balancing algorithm [10] for the private Cloud using virtual machine to physical machine mapping. The architecture of the algorithm contains a central scheduling controller and a resource monitor. The scheduling controller does all the work for calculating which resource is able to take the task and then assigning the task to that specific resource. Ren [11] presented a dynamic load balancing algorithm for cloud computing based on an existing algorithm called WLC (Weighted Least Connection). The WLC algorithm assigns tasks to the node based on the number of connections that exist for that node. This is done based on a comparison of the SUM of connections of each node in the Cloud and then the task is assigned to the node with least number of connections. However, WLC does not take into consideration the capabilities of each node such as processing speed, storage capacity and bandwidth.

The paper in [12] proposes an algorithm called Load Balancing Min-Min (LBMM). LBMM has a three level load balancing framework. It uses the Opportunistic Load Balancing algorithm (OLB) [13]. OLB is a static load balancing algorithm that has the goal of keeping each node in the cloud busy. However, OLB does not consider the execution time of the node. This might cause the tasks to be processed in a slower manner and will cause some bottlenecks since requests might be pending waiting for nodes to be free. LBMM improves OLB by adding a three layered architecture to the algorithm. The first level of the LBMM architecture is the request manager which is responsible for receiving the task.

III. TASK SCHEDULING PROBLEM

The Cloud Computing consists of various size tasks, a collection of interconnected high end resources and a criterion for a performance for scheduling. For this we have taken the Cybershake Scientific Workflow [14] tasks as an input for Cloud Computing system. Table 1 will elaborates the Seismogram Synthesis tasks with its actual weight and expected execution time. These tasks are computationally heavy to execute. Especially, seismogram synthesis tasks are consuming lot of computing resources to execute.
This application is represented by a directed acyclic graph, G=(V,E), where V is the set of v tasks and *E* is the set of *e* edges between tasks. This application's graph has bounded with control-flow dependency constraint such that task t_i should complete its execution before task t_j . In a given application task graph if a task without any parent is call an *entry task* and without having any child task is called as an *exit task.*
We assume that the Cloud Computing data center consist of a set of virtual machines configured by various computing resources such as CPU, memory, bandwidth etc.
We assumed that $et_{i,j}$ gives the estimated execution time to complete the task t_i on Vm_j.
When both ti and t_j are scheduled on the same vitual machine.
The various size of tasks are taking different time to complete its execution on different virtual machine. So, let we describe the task's earliest start time and earliest finish task. $EST(t_i, VM_j)$ and $EFT(t_i, VM_j)$ are the earliest start time of task on VM and earliest finish time of task on VM respectively. The earliest start time for entry task as,

$$EST(t_{entry}, Vm_j) = 0. \qquad (1)$$

For the next tasks in the application, the EST and EFT values are computed in consideration of equ. (1). the subsequent task's start time and finish time have been calculated by equ (2) and equ. (3) respectively.

$$EST(t_i, Vm_j) = \max\left\{ free[j], \max(AFT(tm)) \right\} \qquad (2)$$

$$EFT(t_i, Vm_j) = et_{i,j} + EST(t_i, Vm_j) \qquad (3)$$

The scheduling policy must reduce the length of the waiting queue of tasks for getting the resources. This will possible by solving the objective equ . (4)

$$makespan = \max(AFT(t_{exit})) \qquad (4)$$

This objective function we achieve by our proposed system.

TABLE 1. CYBERSHAKE SEISMOGRAM SYNTHESIS TASKS

Tasks	Size of Tasks (MB)	Time
Task 3	62,69,51,663	39.06
Task 5	69,47,76,323	38.49
Task 7	58,57,63,637	36.27
Task 9	53,68,97,326	32.29
Task 11	67,05,35,542	62.25
Task 14	40,67,28,38,798	96.91
Task 16	45,23,96,996	45.60
Task 18	50,27,64,231	28.67
Task 20	62,41,88,532	24.56
Task 22	42,65,77,006	31.05
Task 24	51,58,32,878	54.87
Task 26	68,14,99,417	23.99
Task 28	44,14,51,516	26.46

IV. PROPOSED DCHEFT

We have developed DCHEFT for Cybershake Scientific Workflow. The fig. 1 explains the architecture of proposed DCHEFT with its mandatory components. Basically, the proposed system has been divided into two main parts. Those are ranking the incoming user's requests (task) and assign that to the resources to minimize the finish time as well as makespan.
The first part completely process the task before it actually assign to the Cloud Computing resources. On the basis of task's estimated execution time, its size and its control flow dependency we assign the rank to every individual task.

We pass the ranked task for processing in waiting queue. As soon as the virtual machine will free in an order to that we assign the task to respective virtual machine. This has been worked out in part two of proposed system architecture.

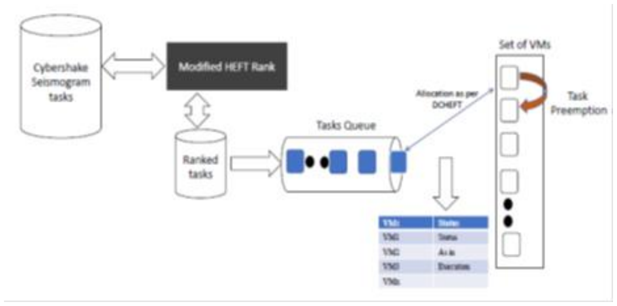

Figure 1. Proposed DCHEFT system architecture

A. *Ranking the tasks by HEFT*

Tasks are order by HEFT algorithm by assigning the first and last ranking. The first rank of a task t_i is calculated by equ. (5).

$$rank_f\,(t_i) = w_i + max_{tj\ e\ succ\ (ti)}\,(c_{i,j} + rank_f\,(t_j)) \qquad (5)$$

where, succ(t_i) is the successors of task t_i , $c_{i,j}$ is the average communication cost of edge (i, j) and w_i is the average computation of task t_i . The first task's which is ready to schedule on virtual machine its first rank is equal to

$$rank_f\,(t_{exit}) = w_{exit} \qquad (6)$$

The last rank of a task t_i is calculated as follows.

$$rank_l\,(t_i) = max_{tj\ e\ predc\ (ti)}\,(rank_l\,(t_j) + w_j + c_{i,j}) \qquad (7)$$

where, pred(t_i) is the predecessor of task t_i . The last ranks are computed recursively by traversing the task graph towards the last task of an application which has been started from the first task of the graph.

B. *Proposed Divide and Conquer approach*

In the HEFT the second phase is to schedule the ranked task to virtual machine. While allocating the tasks to the virtual machine the HEFT has considered the idle time of virtual machines. An idle time is the difference between execution start time and finish time of two tasks that were scheduled on the same virtual machine.

However HEFT has some limitations. HEFT algorithm search for an idle time and if idle time is less than the scheduled task's execution time then task must have to wait until next idle time. This HEFT affected on the waiting time of tasks. This is the major motivation behind the work presented in this paper. The proposed divide and conquer based HEFT algorithm initially finds the idle time and schedule the task without consideration whether idle time is less than task's execution time. Fig. 2 explains the detail flow of the proposed DCHEFT system.

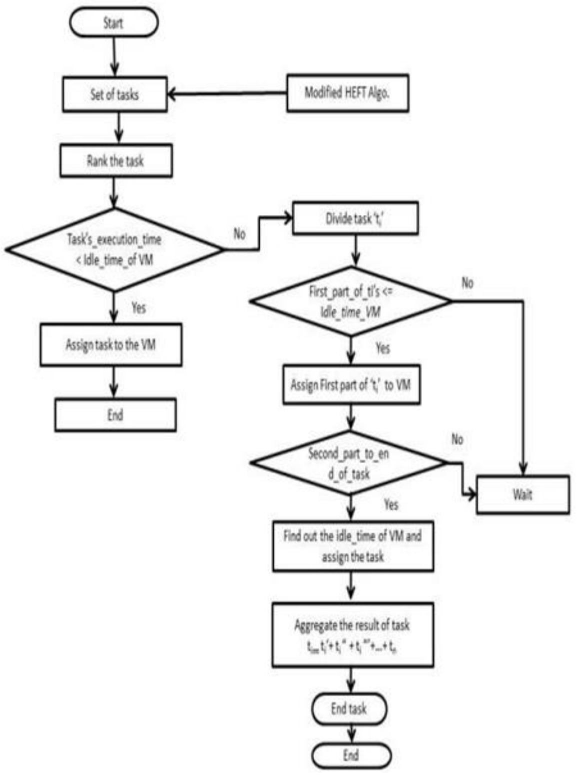
Start
Set of tasks
Modified HEFT Algo.
Rank the task
Task's_execution_time < Idle_time_of VM
No
Yes
Assign task to the VM
End
Divide task 't_i'
First_part_of_ti's <= Idle_time_VM
No
Yes
Assign First part of 't_i' to VM
Second_part_to_end_of_task
No
Yes
Wait
Find out the idle_time of VM and assign the task
Aggregate the result of task t_i = t_i' + t_i'' + t_i''' + ... + t_n
End task
End

Figure 2. Proposed System Flowchart

```
Algorithm: DCHEFT
Input- Task's Execution Time, Task $t_i$
Output- Task's Output
1: Start
2: ti's Execution time is 'x' then
3: If(x< idle_time_of_virtual_machine)
4:         Assign 'x' to that Virtual_Machine;
5: Else
6:         Divide 'x';
7:         If (First part 'x' <= idle_time)
8:                 Assign 'x' to Virtual_machine;
9:         Else
                   Wait for Ideal-Time;
10:        For second part to end of the part of task
11:        Do
12:                find-out the idle-time of VM and
                   assign to the task;
13:        enddo
14:        endfor
15:        Aggregate the execution of $x= x`+x``+…+x`_n$;
16: End
```

V. Experimental Setup

The proposed DCHEFT approach work is experimented on Cloud Simulator [15], which gives the real-time environment scenario of Cloud Computing. Datacenter Information has been listed in Table 2. Tables 3 consist of configuration for Datacenter which includes allocation policy, architecture, OS, hyper visor, scheduling and monitoring interval, threshold value etc. Host in the Datacenter used to show the amount of provisional RAM, bandwidth, storage capacity, power, processing element etc. of given task which process by Datacenter . Table 4 explains the host configuration details. Configuration details of customized simulation setup are given in Table 5 and it consist of general information of Datacenters like number of Datacenters, number of host, number of processing units, capacity etc. Every Datacenter component instantiates a generalized application provisioning component that implement a set of policies for allocating bandwidth, memory and storage devices to hosts and virtual machines. Table 6 holds information related to storage area network capacity, latency and bandwidth.

TABLE 2: DATACENTER INFORMATION

Sr. No.	Information	Contains
1	Number of Datacenter	1
2	Number of Host	1
3	Number of Processing Units	4
4	Processing capacity (MIPS)	9600
5	Storage Capacity	11 TB
6	Total Amount of RAM	40 GB

TABLE 3: DATACENTER CONFIGURATION DETAILS

Sr. No.	Information	Contains
1	Allocation Policy	SDMCOA
2	Architecture	X86
3	Operating system	Linux
4	Hypervisor	Xen
5	Upper threshold	0.8
6	Lower threshold	0.2
7	VM Migration	Enabled
8	Monitoring Interval	180

TABLE 4: HOST CONFIGURATION DETAILS

Sr. No.	Information	Contains
1	RAM	40 GB
2	Bandwidth	10,00,000
3	Operating System	Linux
4	Hypervisor	Xen

TABLE 5: CUSTOMER CONFIGURATION DETAILS

Sr. No.	Information	Contains
1	Users	1
2	Cloudlets sent per minutes	50
3	Avg. Length of Cloudlet	50,000
4	Avg. Cloudlet file Size	500 Bytes
5	Avg. Cloudlet output size	500 Bytes

TABLE 6: CUSTOMER CONFIGURATION DETAILS

Sr. No.	Information	Contains
1	Number of VMs	20
2	Avg. Image Size	1000 Bytes
3	Avg. RAM	512 MB
4	Avg. Bandwidth	1,00,000 Mbps
5	Procedure Element	1
6	Priority	1
7	Scheduling Priority	Dynamic Workload

VI. Result And Discussion

This section will brief about the performance of proposed novel DCHEFT approach.

Let, we evaluate our proposed DCHEFT approach with existing BATS [16] and Heuristic Approach [17], SDMCOA [18] on the given Cybershake Seismogram Synthesis tasks. Evaluation of proposed DCHEFT system is based on two key factors i.e. turnaround time and response time.

A. *Evaluation of Turn Around Time*

This is one of the major performance factor to check the evaluation of the system. Basically, it is the span of total time taken between the submission of a request (task) for execution to the complete the same. Specifically, turnaround time is based up on the programming/ software logic. We compare our proposed DCHEFT system with BATS, Heuristic and SDMCOA. We found that our system is works fine as compared with existing systems which has been shown result in Table 7 and Fig. 3.

TABLE 7: TURN AROUND TIME COMPARISON IN MS

Tasks	DCHEFT	SDMCOA	Heuristic	BATS
Task 3	2405.79	2613.79	2832.94	3599.29
Task 5	2405.07	2613.07	2914.42	3599.29
Task 7	2403.02	2611.02	2913.87	3599.29
Task 9	2398.93	2606.93	2911.75	3599.29
Task 11	2498.78	2636.78	2907.67	3599.29
Task 14	2348.36	2556.36	2772.11	3599.29
Task 16	2346.44	2554.44	2857.89	3599.29
Task 18	2329.48	2537.48	2855.97	3599.29
Task 20	2325.36	2533.36	2833.36	3599.29
Task 22	2332.06	2540.06	2834.72	3599.29
Task 24	2355.70	2563.70	2841.49	3599.29
Task 26	2324.86	2532.86	2832.86	3599.29
Task 28	2327.37	2535.37	2833.96	3599.29

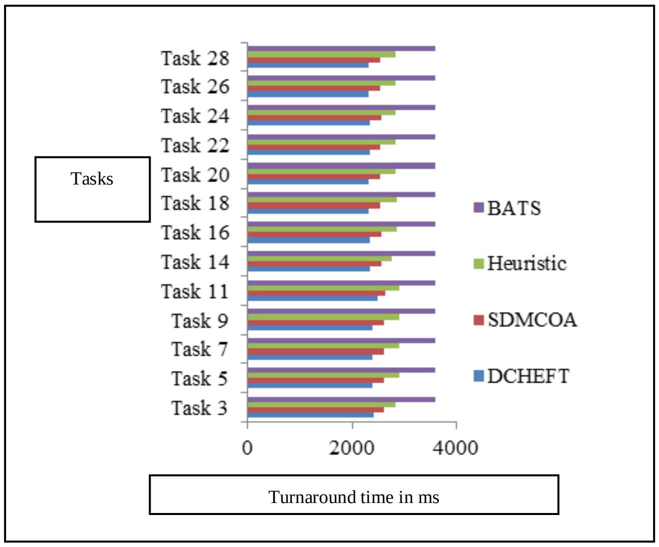

Figure 3 TAT Comparison in ms

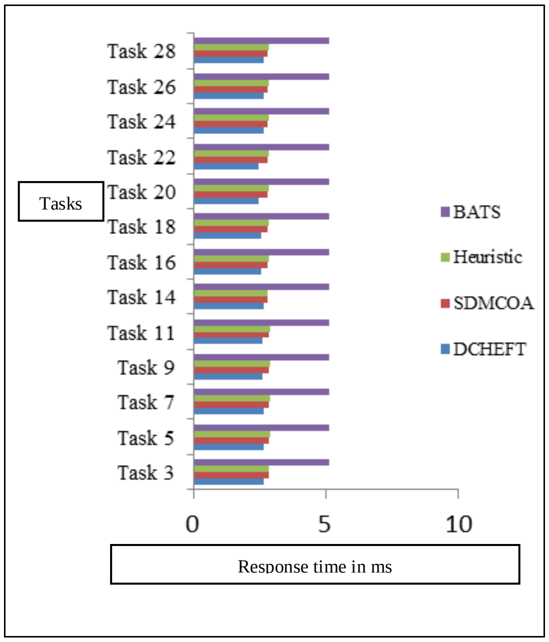

Figure 4 RT Comparison in ms

B. *Evaluation of Response Time*

This is another major performance factor to check the evaluation of the system. Response time is the time taken from the issuance of a task to the commence of a response to that task. We compare our proposed DCHEFT system with BATS, Heuristic and SDMCOA. We found that our system is works fine as compared with existing systems which has been shown result in Table 8 and Fig. 4.

TABLE 7: RESPONSE TIME COMPARISON IN MS

Tasks	DCHEFT	SDMCOA	Heuristic	BATS
Task 3	2.63	2.83	2.83	5.1
Task 5	2.63	2.83	2.91	5.1
Task 7	2.63	2.83	2.9	5.1
Task 9	2.59	2.83	2.91	5.1
Task 11	2.59	2.83	2.90	5.1
Task 14	2.63	2.77	2.77	5.1
Task 16	2.54	2.77	2.85	5.1
Task 18	2.54	2.77	2.85	5.1
Task 20	2.46	2.77	2.83	5.1
Task 22	2.46	2.77	2.83	5.1
Task 24	2.63	2.77	2.84	5.1
Task 26	2.63	2.77	2.83	5.1
Task 28	2.63	2.77	2.83	5.1

VII. CONCLUSION

This paper describes a proposed Divide and Conquer Heterogeneous Earliest Finish Time Algorithm for task scheduling to efficiently managed the resources in Cloud Computing. To utilize the resources ideal time we used the concept of Divide and Conquer. Because of this methodology the task's waiting time is drastically reduce while the utilization of resources have been increased which has been proved by experimentally. The results from various simulations using Cybershake Scientific Seismogram tasks as an input shows that the DCHEFT approach performs better than SDMCOA, Heuristic and BATS existing approaches. In future the tasks waiting time needs to be reduced when the availability of resources are less.

REFERENCES

[1] Zhang Qi, Lu Cheng, and Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges", *Journal of internet services and applications*, no. 1 (2010): pp 7-18.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Amazon AWS, http://aws.amazon.com/. Accessed online 02 June 2017. K. Elissa, "Title of paper if known," unpublished.

[3] Microsoft cloud, http://www.microsoft.com/en-us/cloud/.Accessed online 02 June 2017..

[4] IBM cloud, http://www.ibm.com/ibm/cloud/. Accessed online 02 June 2017..

[5] Copcuoglu, Haluk, Salim Hariri, and Min-you Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE transactions on parallel and distributed systems 13.3 T(2002): 260-274.

[6] Google Compute Engine, https://cloud.google.com/solutions/reliable-task-scheduling-compute-engine. Accessed Online 04 June 2017.

[7] H. Liu, D. Orban, GridBatch: cloud computing for large-scale data-intensive batch applications, in: IEEE International Symposium on Cluster Computing and the Grid, pp. 295–305, 2008.

[8] Niels Fallenbeck, Hans-Joachim Picht, Matthew Smith, and Bernd Freisleben. 2006. Xen and the Art of Cluster Scheduling. In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC '06). IEEE Computer Society, Washington, DC, USA, 4-. DOI=http://dx.doi.org/10.1109/VTDC.2006.18.

[9] M. Wilde, M. Hategan, J.M. Wozniak, B. Clifford, D.S. Katz, I. Foster, Swift: a language for distributed parallel scripting, Parallel Computing 37 (9) (2011) 633–652.

[10] Ni, J., Y. Huang, Z. Luan, J. Zhang and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment," in proc. International Conference on Cloud and Service Computing (CSC), IEEE, pp: 292-295, December 2011.

[11] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.

[12] Wang, S-C., K-Q. Yan, W-P. Liao and S-S. Wang, "Towards a load balancing in a three-level cloud computing network", in proc. 3rd International Conference on. Computer Science and Information Technology (ICCSIT), IEEE, Vol. 1, pp:108-113, July 2010.

[13] Sang, A., X. Wang, M. Madihian and RD. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, Vol. 14, No. 1, pp: 103-120, January 2008.

[14] Pegasus Scientific Workflow. https://confluence.pegasus.isi.edu/display/pegasus/ WorkflowGenerator. Access Online. 04 June 2017.

[15] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), pp.23-50, 2011.

[16] Mahendra Bhatu Gawali and Subhash K. Shinde. "Implementation of IDEA, BATS, ARIMA and queuing model for task scheduling in cloud computing." In *Eco-friendly Computing and Communication Systems (ICECCS), 2016 Fifth International Conference on*, pp. 7-12. IEEE, 2016. DOI: 10.1109/Eco-friendly.2016.7893233.

[17] Mahendra Bhatu Gawali and Subhash K. Shinde (Under Review) "Task Scheduling and Resource Allocation in Cloud Computing by Heuristic Approach". Springer, *Journal of Cloud Computing, Advances, Systems and Applications.* DOI 10.1186/s13677-017.

[18] Mahendra Bhatu Gawali and Subhash K. Shinde, "Standard Deviation Based Modified Cuckoo Optimization Algorithm for Task Scheduling to Efficient Resource Allocation in Cloud Computing", *Journal of Advances in Information Technology, Vol. 8, No. 4, pp. 210-218, November, 2017.* doi: 10.12720/jait.8.4.210-218.

AUTHOR PROFILE

Mahendra Bhatu Gawali received his BE degree in 2008 and M.E. degree in 2013 from North Maharashtra University, Jalgaon, MS, India. Currently he is pursuing his Ph.D. at Thadomal Shahani Engineering College, Bandra(W), University of Mumbai, Mumbai, India. He focuses on Task Scheduling and Resource Allocation in Cloud Computing.

Subhash K. Shinde is working as a Professor in the Department of Computer Engineering at Lokmanya Tilak College of Engineering, Navi Mumbai, India. He received his Ph.D. from Swami Ramanand Teertha Marathwada University, India in 2012. He has published more than 40 research papers in the field of Web Mining, Frequent Pattern Discovery and Integration of domain knowledge in web personalized recommendations in the reputed journals and conferences.